Lab 7 - Code Design I

Saturday, April 18, 2020 10:30 AM

Pseudocode design to implement functions

Initialization:

- Initialize timer		Same as previous labs
- Initialize LED ports		Same as previous labs
- Initialize BTN ports		Same as previous labs

- ---
- Initialize SPI Interface
    - Reset LCD Display

```
#define SPI_BRG 31          // value of BRG for 156.25kHz with
                            // peripheral clock at 10Mhz

 unsigned long int spi1mode1;       // SPI Configuration variables
 unsigned long int spi1mode2;

  // clear the SPI SFRs before configuring for use
  SPI1CON = 0;     // Reset SPI2 configuration register
  rData = SPI1BUF;       // clear the receive buffer
  SPI1BRG = SPI_BRG;     // set brg register value
  SPI1STATCLR=0X40;   // clear the overflow

  // Configure SPI2 using the c peripheral library functions

  spi1mode1 = MASTER_ENABLE_ON | SPI_SMP_ON | SPI_CKE_ON |
          SPI_MODE8_ON;
  spi1mode2 = SPI_ENABLE;
  OpenSPI1(spi1mode1, spi1mode2);

  /****************** Reset LCD Display Controller **************/

       PORTClearBits (IOPORT_D, BIT_9);
       DelayMs(100);
       PORTSetBits (IOPORT_D, BIT_9);
       DelayMs(500);

 /*************** LCD Display initialization commands ***************/

  // LCD module reset command sequence
  SPI1BUF=0x1b;                  // Display reset - first send escape char
  c_buffer = "[*";               // command sequence for reset
  putsSPI1(2,c_buffer);   // write out string
  DelayMs(500);           // wait for display to reset
```

```
// LCD module cusor command sequence
SPI1BUF=0x1b;                   // Cursor reset - first send escape char
c_buffer = "[0c";               // command sequence for cursor off
putsSPI1(3,c_buffer);   // write out string
DelayMs(500);           // wait for display to reset

// LCD module wrap command sequence - use 40 to eliminate conflicst with lab reqs
SPI1BUF=0x1b;                   // Display reset - first send escape char
c_buffer = "[1h";               // command sequence for 40 char wrap
putsSPI1(3,c_buffer);   // write out string
DelayMs(500);           // wait for display to reset


      putsSPI1(16, buffer);                     // Diagnostic
      DelayMs(500);
```

- ---

- Initialize I2C Interface
  - Configure TMP3 module

```
#define I2C_BRG 0x009                   // baud rate for 10Mhz clock to I2C
                                        // for 100Khz fsck


 unsigned char i2c_data[4];                             // buffer for TMP3 data
unsigned char SlaveAddress;                     // I2C slave device TMP3 address
int Index;                                      // I2C communication variables
int DataSz;



/* ---- configure and initialize the I2C interface ---- */
 OpenI2C1(I2C_EN, I2C_BRG);   // configure I2C module 2 - mostly default options

/* ---- initialize tmp3 module to read 9 bit temp ----- */
SlaveAddress = 0x4f;     // slave address of TMP3 pmod set with
                                        // jumpers JP3,JP2,JP1
i2c_data[0] = (SlaveAddress << 1) | 0x0; // address with write bit set
i2c_data[1] = 0x00;             // select register 0 on tmp3 module
DataSz = 2;                             // specify two bytes for register config message

/* --- transmit command to TMP3 module via I2C ----  */
StartI2C1();                            // send the start bit
IdleI2C1();                                     // wait to complete
 Index = 0;       // transmit bytes to tmp3 module

while( DataSz )
  {
      MasterWriteI2C1 (i2c_data[Index++]);
      IdleI2C1();               // wait to complete
      DataSz--;
      if(I2C1STATbits.ACKSTAT)          //ACKSTAT is 0 on slave ack
            break;
  }
StopI2C1();                             // Stop I2C bus
 IdleI2C1();                            // wait to complete
```

- Initialize UART Interface

```
unsigned long int u2mode;           // UART Configuration variables
unsigned long int u2status;


/**************    Initialize UART ***************************/

 u2mode = UART_EN | UART_IDLE_CON | UART_RX_TX |
        UART_DIS_WAKE | UART_DIS_LOOPBACK |
        UART_DIS_ABAUD | UART_NO_PAR_8BIT | UART_EN_BCLK |
        UART_1STOPBIT | UART_IRDA_DIS | UART_MODE_SIMPLEX |
        UART_NORMAL_RX | UART_BRGH_SIXTEEN;
   u2status = UART_TX_PIN_LOW | UART_RX_ENABLE | UART_TX_ENABLE;

        // configured for 9600 baud - 8N1
        OpenUART2(u2mode, u2status, 64);
 WriteUART2(62 );                // user prompt
 WriteUART2(13 );                // carraige return
 WriteUART2(10 );                // line feed
```